June 5, 1989 Final

X.25-ISDN Gateway Program Optimization Results

C-DCA100-87-C-0063

Booz-Allen & Hamilton, Inc.
4330 East West Highway
Bethesda, MD 20814

National Communications System
Office of Technology & Standards
Washington, DC 20305-2010

NCS TIB 89-5

Point of Contact: Dennis Bodson, 202-692-2124



Present telecommunications networks use one of many different protocol suites to interconnect their services. The networks employing different protocols cannot maintain communications with one another because the diverse protocols are incompatible. To address the protocol imcompatibility issues, it was recommended to develop mechanisms, such as gateways, to interconnect networks using these diverse protocols. These devices will act as translators between the two networks while maintaining communications with each network in its native language.

ISDN Gateways 57
National Security Emergency Preparedness (NSEP)

Unclassified Unclassified Unclassified

NATIONAL COMMUNICATIONS SYSTEM

TECHNICAL INFORMATION BULLETIN

89-5

# X.25-ISDN GATEWAY PROGRAM OPTIMIZATION RESULTS



JUNE 5, 1989

OFFICE OF THE MANAGER
NATIONAL COMMUNICATIONS SYSTEM
WASHINGTON, D.C. 20305

# X.25-ISDN GATEWAY PROGRAM OPTIMIZATION RESULTS

JUNE 5, 1989

BOOZ·ALLEN & HAMILTON INC.
4330 EAST WEST HIGHWAY
BETHESDA, MD 20814

PREPARED FOR
THE OFFICE OF THE MANAGER
NATIONAL COMMUNICATIONS SYSTEM
8TH & SOUTH COURTHOUSE ROAD
ARLINGTON, VA 22204

# TABLE OF CONTENTS



# L I S T  O F  E X H I B I T S

# 1.0 INTRODUCTION

The Office of the Manager, National Communications System (CS), has been tasked to ensure the survivability of the onal telecommunications infrastructure in emergency itions, including nuclear attack. In support of this task, OMNCS is working to promote standards within the data unication and telecommunication fields to increase the roperability of diverse communication systems. This oves the National Security Emergency Preparedness (NSEP) unication capabilities of the nation by providing additional rconnectivity among Federal telecommunication assets.

## BACKGROUND

Present telecommunication networks use one of many different ocol suites to interconnect their services. The networks oying different protocols cannot maintain communications one another because the diverse protocols are mpatible. To address the protocol incompatibility issue, OMNCS recommended developing mechanisms, such as gateways, nterconnect networks using these diverse protocols. These ces will act as translators between the two networks while taining communications with each network in its native uage. In addition, the OMNCS has recognized the upcoming sitions that are needed to support new and evolving nology such as the integration of voice and data services digital networks.

In the data communications field there are a number of alent standards for supporting end-to-end communications. e these standards are expected to co-exist during the near re, gateways are needed that will allow users of these rse protocols to communicate.

Recent efforts have focused on developing a network onent to allow X.25 end-user traffic to use future ISDNs as sport networks to interconnect disrupted portions of their et-switched networks. These efforts are determining the ibility of using excess the transmission capacity of ISDN of-band signaling channels. To assess this capability, a arison of the two protocols was prepared. This was followed a definition of protocol operating states and transitions g states needed to control and implement an X.25-ISDN way. Based on these preliminary operating states, a SDL ification for a X.25-ISDN Gateway was developed. The -ISDN Gateway has been implemented and its capabilities nstrated. The X.25-ISDN Gateway and demonstration is ribed in X.25-ISDN Gateway High Level Language Program nstration, National Communications System, May 12, 1989.

## 1.2 PURPOSE

The purpose of this document is to:

- Present the optimized software component of the gateway
- Document gateway software
- Discuss potential enhancements.

## 1.3 ORGANIZATION

This X.25/ISDN Gateway Program Optimization Report is comprised of four sections. Section 2.0 describes the gateway software design. Detailed process specifications are included in this section.

A listing of the gateway software code is contained in section 3.0.

Section 4.0 summarizes the document and discusses some major issues involving the X.25/ISDN Gateway. The next steps to be taken are also discussed.

# 2.0 GATEWAY SOFTWARE DESIGN

The gateway was designed to be portable and modular. Written in C language, the program allows the user to boot the TELEOS and the EICON boards, and examine statistics from the gateway, the ISDN Network Basic Input/Output System (NETBIOS), or the X.25 Network Adapter BIOS (NABIOS). The board interface software is commonly known as the Network Basic I/O System interface or NETBIOS. For the gateway's two boards, there are two NETBIOS interfaces, NABIOS for the Eicon X.25 board and NETBIOS for the Teleos B100PC ISDN board. In addition, the user can control packet transmission, routing, and call clearing.

The gateway switch architecture is composed of input/output (I/O) components and their software control modules. The components allow the external influences (the user, X.25, and ISDN boards) to communicate to the internal gateway software environment or modules. Exhibit 3-1 shows the relationship between the components and the gateway. Each of the components is discussed below.

## 2.1 Man-Machine Interface

The MMI allows the user sitting at the gateway's PC to control the execution of the gateway. The interface allows the user to use the PC keyboard to signal for initialization of the X.25 and ISDN boards, display gateway packet transfer statistics, start-up the gateway connections, and shutdown of all outstanding connections.

## 2.2 Statistics Gathering

Statistics gathering occurs in two ways. First, the gateway provides internal statistics of all packet transactions from one side to the other. Second, the gateway calls for the statistics from either of the two external NETBIOS software modules that reside on the PC. The NETBIOS statistics return actual protocol-specific metrics of packet transfer activity.

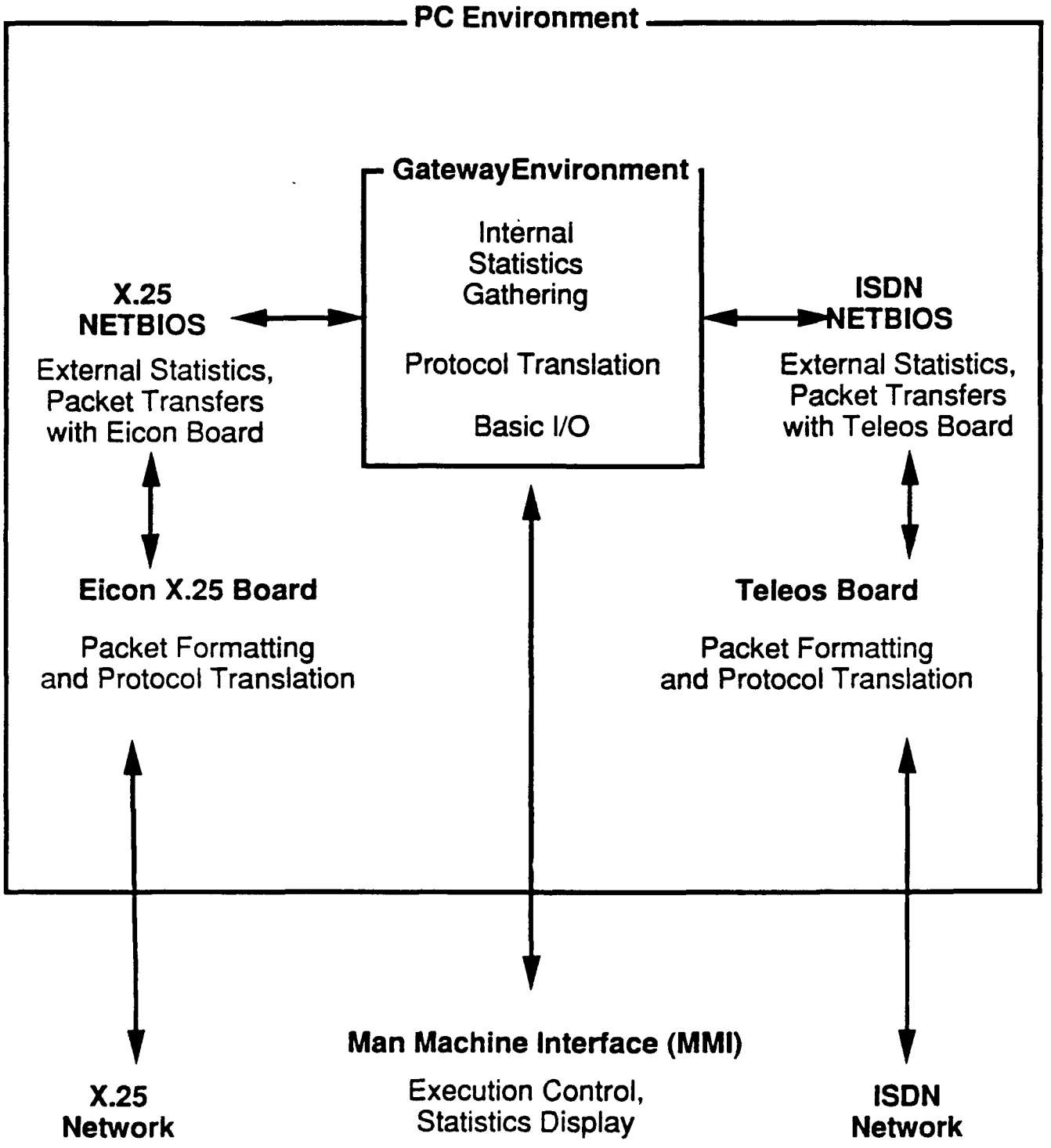
## 2.3 Protocol Translation

The major protocol translation function that takes place in the gateway is address translation. When a call request arrives, the gateway performs a lookup in the Internal Address Translation Matrix using the incoming destination address as the key. A destination address that is valid for the opposite side's protocol is mapped to the subsequent outgoing call request.

## 2.4 Basic I/O

In the realm of the PC, an incoming packet cannot be sent to

Exhibit 2-1
Gateway Environment



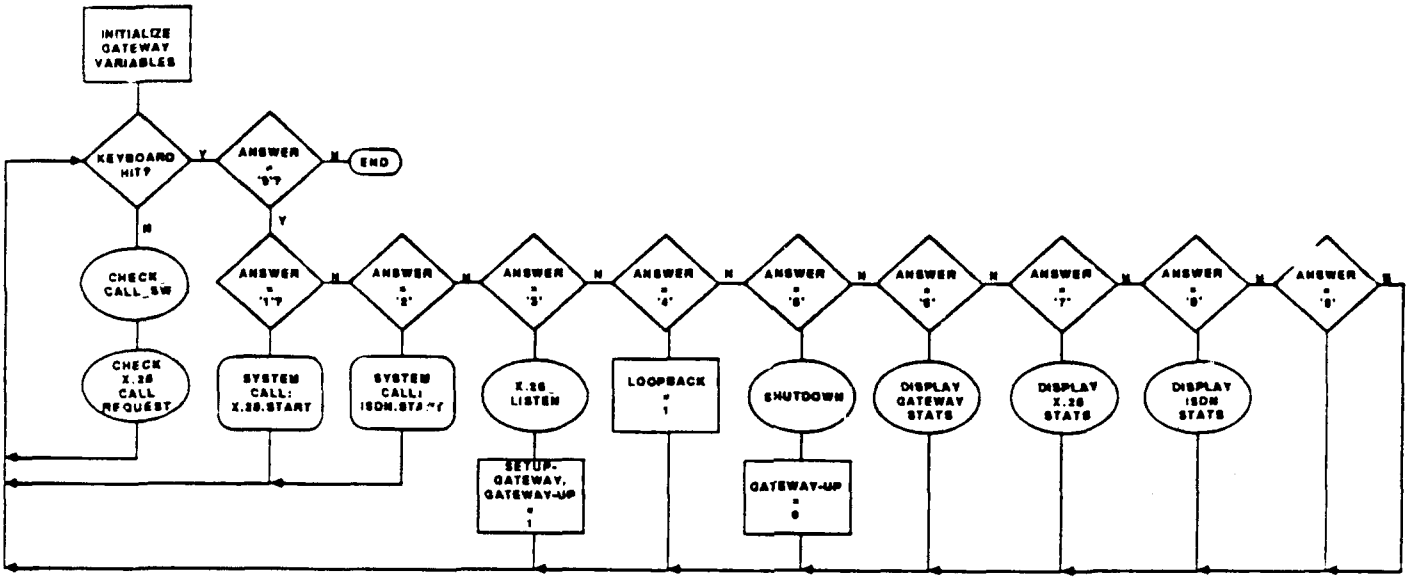
the output port of the destination network. Instead, packets must pass from the network link through the protocol's PC board, then through the board interface software to the gateway. Similarly, outgoing packets are passed from the board interface software to the PC network board, to the destination network.

To access the respective NETBIOS so the packets can be passed, the gateway must build a NETBIOS Control Block (NCB). Each NCB must contain a command code defining the action that NETBIOS must tell the board to perform. In addition, each NCB must have several fields filled with board-specific information detailing which board software routine to use and what logical session number to reference. NCBs must contain the address and length of the data being passed. Once the proper fields have been filled, the gateway issues a system interrupt, which passes the address of the NCB to the correct NETBIOS.

With the exception of the NETBIOS interfaces, all of the preceding components are implemented internally in the gateway. The NETBIOS software comes with the standard distribution package from the board vendors.

The detailed process specifications are included at the end of this section. Each figure depicts the processing flow of a module within the gateway including all software calls to other routines.

# EXHIBIT 2-2
# GATEWAY: MAIN



# EXHIBIT 2-3
# CHECK_CALL_SW
# (*NCB_PTR1, *NCB_PTR2, *NCB_PTR3, *NCB_PTR4)

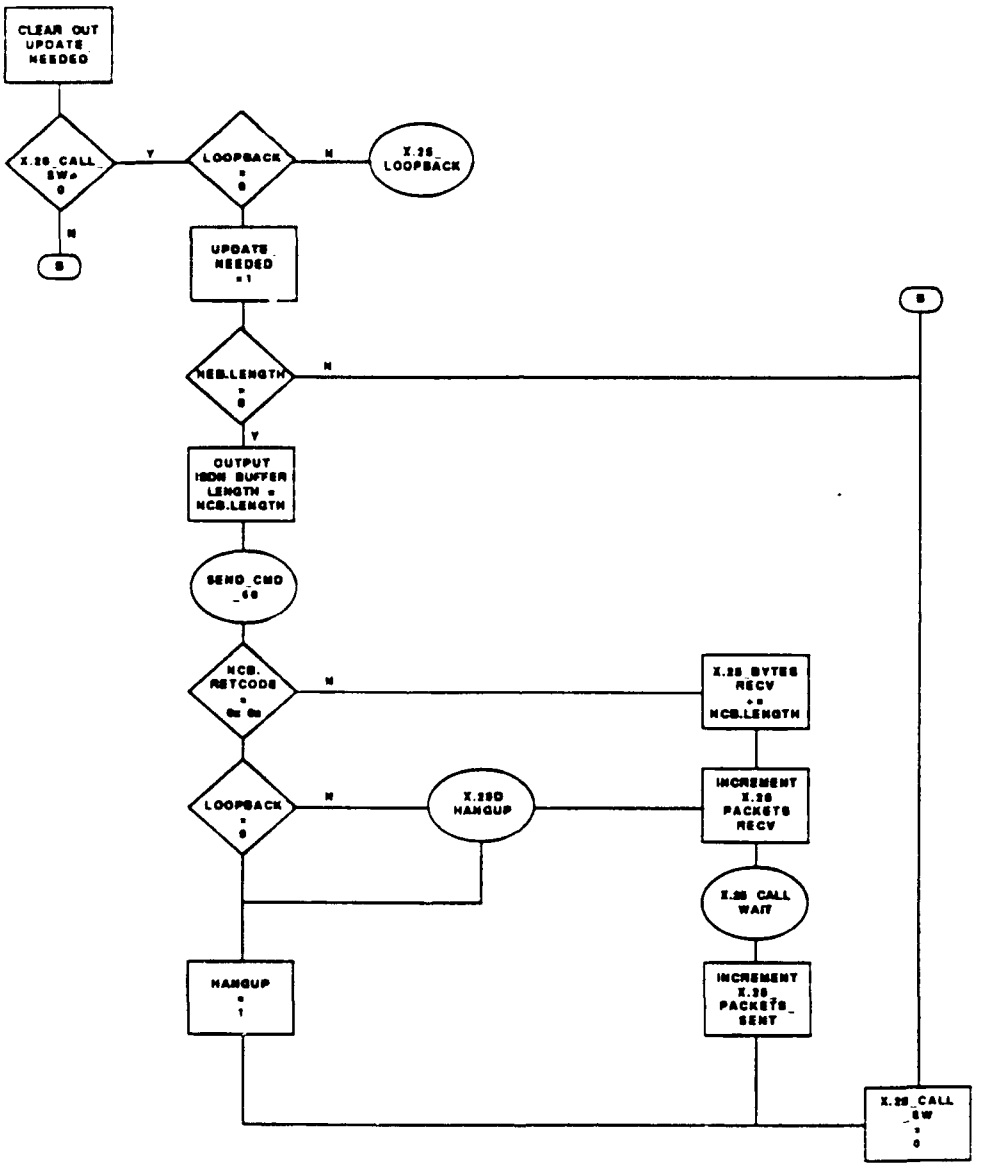


EXHIBIT 2-4
CHECK_CALL_SW
(*NCB_PTR1, *NCB_PTR2, *NCB_PTR3, *NCB_PTR4)
(CONT.)

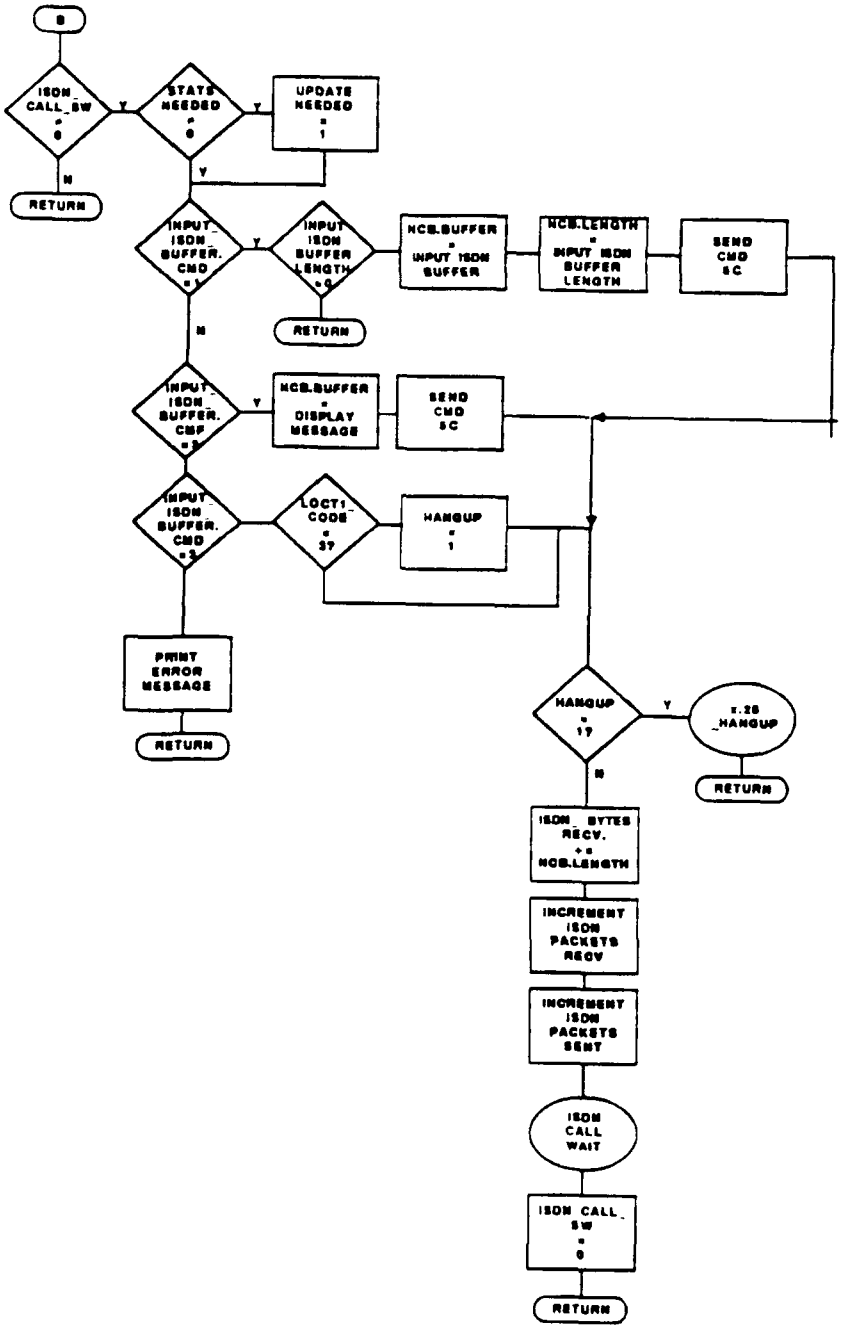


EXHIBIT 2-5
CLEAR_NCB
(*NCB_POINTER)

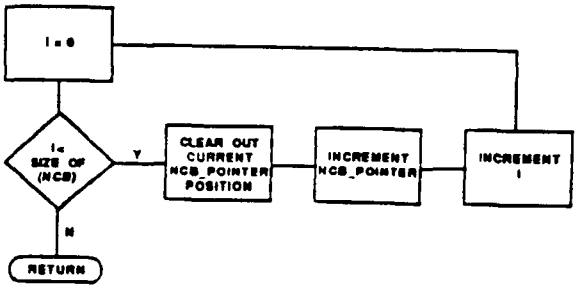


EXHIBIT 2-6
SETUP
(*NCB_PTR1, *NCB_PTR2, *NCB_PTR3, *NCB_PTR4)

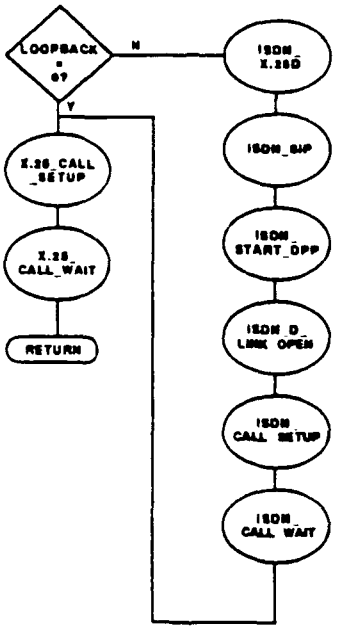


EXHIBIT 2-7
X.25_LISTEN
(*NCB.POINTER)

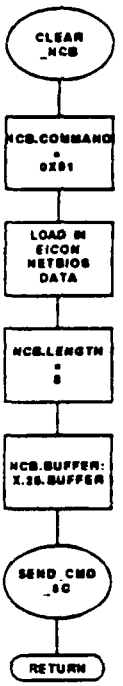


EXHIBIT 2-8
ISDN SEND

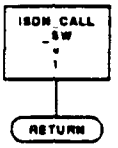



EXHIBIT 2-9
X.25 SEND

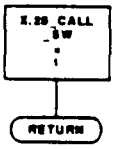


EXHIBIT 2-10
ISDN_X.25D

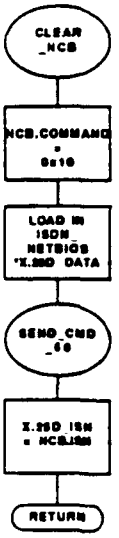
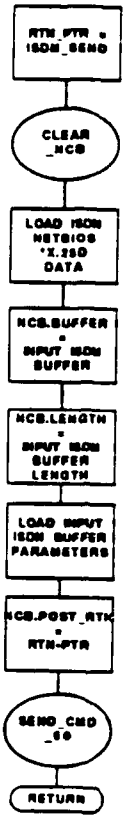


EXHIBIT 2-11
ISDN CALL WAIT
(*NCB_POINTER)



# EXHIBIT 2-12
# ISDN_SIP

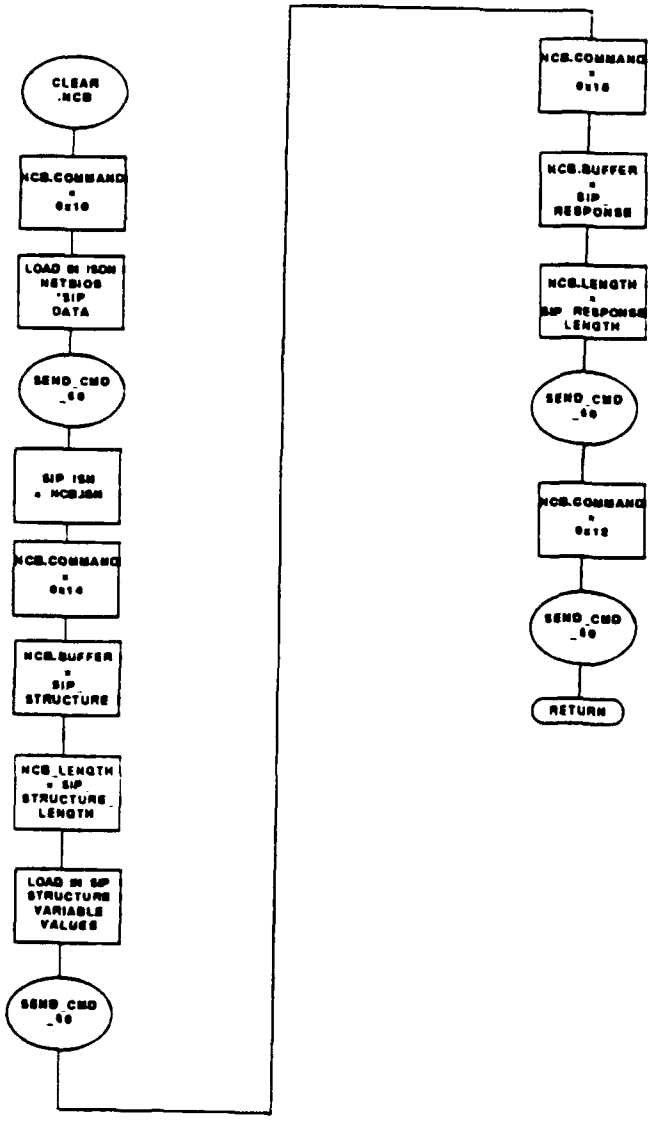
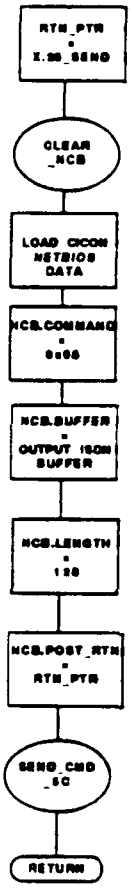


EXHIBIT 2-13
ISDN_CALL_WAIT
(*NCB_POINTER)



# EXHIBIT 2-14
# ISDN_START_DPP

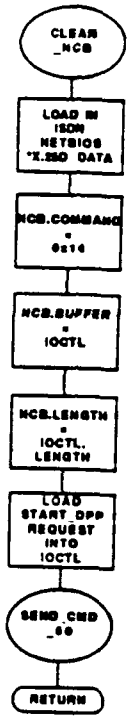
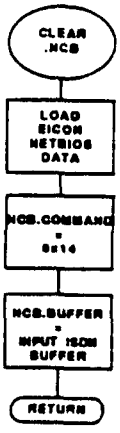


EXHIBIT 2-15
CALL_SETUP



# EXHIBIT 2-16
# ISDN_D_LINK_OPEN

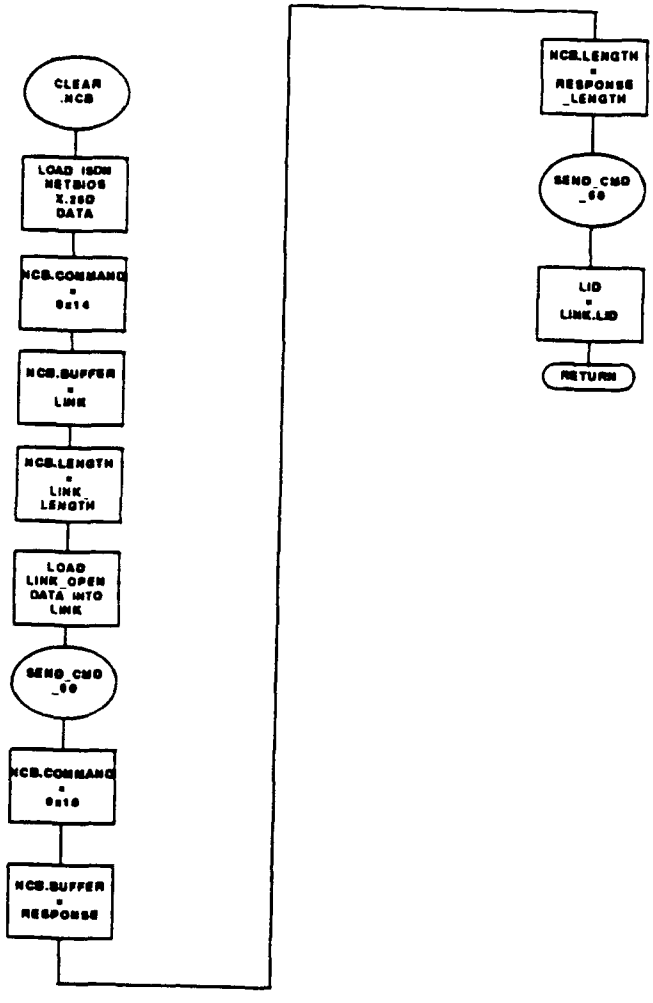


EXHIBIT 2-17
ISDN CALL SETUP
(*NCB_POINTER)

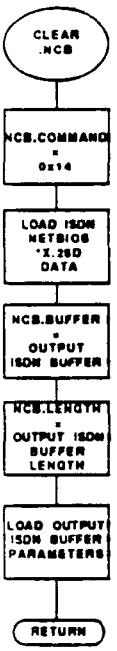


EXHIBIT 2-18
X.25_HANGUP

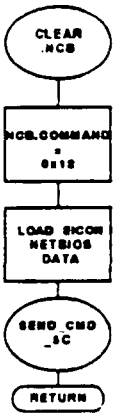


EXHIBIT 2-19
X.25D_HANGUP

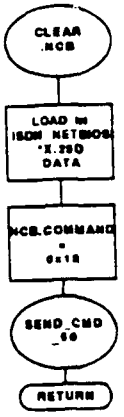
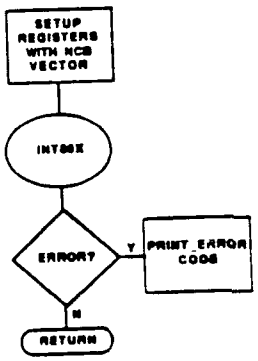
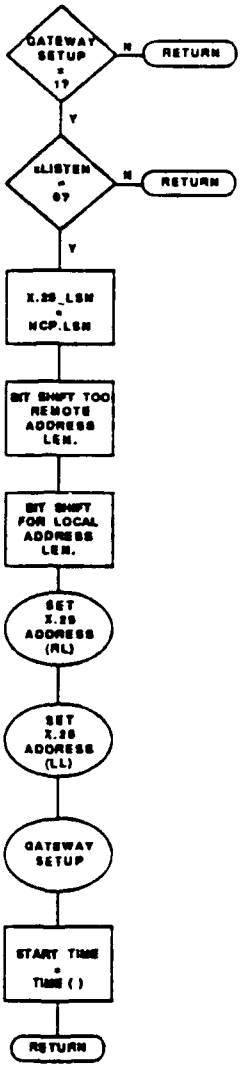


EXHIBIT 2-20
SEND_CMD_5C
(*NCB_POINTER)



# EXHIBIT 2-21
# CHECK_X.25_CALL_REQUEST



# EXHIBIT 2-22
# SET_X.25_ADDRESS
# (LENGTH, X.25_BUFFER ADDRESS, X.25 NCB BUFFER)

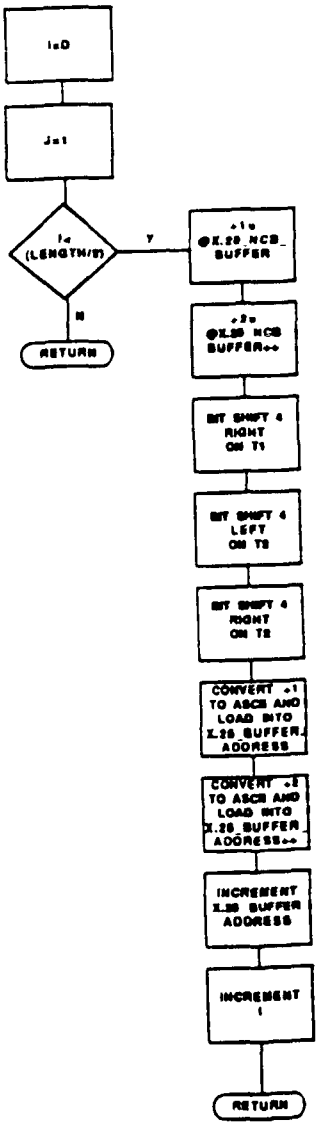


EXHIBIT 2-23
SHUTDOWN

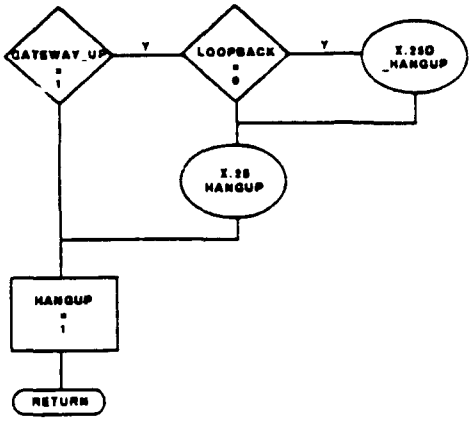
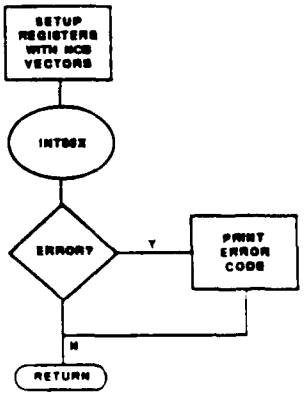



EXHIBIT 2-24
SEND_CMD_60
(*NCB_POINTER)



## 3.0 GATEWAY SOFTWARE CODE

The code presented is the implementation of the detailed design shown in Section 2.0. The code was developed in the "C" programming language using Microsoft Quick C V1.1 and Microsoft Quick C Compiler for the DOS environment.

The gateway program is a set of modular I/O calls that allow the user to connect between X.25 and ISDN networks. Since being developed and run at the C&P site in Richmond, Va., the gateway has been modified to work with other ISDN switches. At the BAH System Resource Center in Bethesda, the gateway and its associated software has been tested and demonstrated using our Teleos IAP ISDN switch. What follows is a complete listing of the gateway in the C programming language.

```
#include <dos.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <graph.h>
#include <math.h>

#define X25_NAME          "X25               "
#define X25_DIAL_CODE     "03141"
#define NB_NAME_LEN       16
#define NB_NCB_RESERVE    14
#define X25D_NAME         "*X25D             "
#define SIP_NAME          "*SIP              "
#define NETBIOS_60        ((unsigned char) 0x60)
#define NETBIOS_5C        ((unsigned char) 0x5c)
#define OK                0
#define ESCAPE            0x1b

typedef unsigned char uchar;
typedef unsigned short ushort;
typedef unsigned long ulong;
/*
 NCB Structure
*/
typedef struct
{
          uchar ncb_command;
          uchar ncb_retcode;
          uchar ncb_lsn;
          uchar ncb_num;
          uchar far *ncb_buffer;
          ushort ncb_length;
          uchar ncb_callname[NB_NAME_LEN];
          uchar ncb_name[NB_NAME_LEN];
          uchar ncb_rto;
          uchar ncb_sto;
          ushort ui1;                          /* Segment address word */
          ushort ui2;                          /* Offset address word  */
          uchar ncb_lana_num;
          uchar ncb_cmd_cplt;
          uchar ncb_reserve[NB_NCB_RESERVE];
} ncb_t;
union
{
          struct data_message_type
          {
                    short data_cmd;
                    short data_LID;
                    short data_Control;
                    short data_Size;
                    char data_Data[128];
          }dm;
          struct display_type
          {
```

```
                short dis_cmd;
                char dis_Message[128];
        }display;
        struct control_type
        {
                short ioctl_cmd;
                short ioctl_CODE;
                short ioctl_Size;
                char ioctl_Data[128];
        }ioctl;
}iu,ou;

union  REGS regs;
struct SREGS sregs;
char            error_message [] = "This video mode is not supported";
char            obuf[80];
char            buffer[255];
char            x25_buffer[128];
char            call_buffer[7]={2,0x8a,0,1,0,0,0};
char            menu_choice[10]={0,0,0,0,0,0,0,0,0,0};
char            x25_send_address[20];
char            x25_recv_address[20];
char            isdn_send_address[20];
char            isdn_recv_address[20];
uchar           *srcp;
uchar far       *destp;
short           sw;
short           lid;
short           x25_call_sw;
short           isdn_call_sw;
uchar           x25_lsn;
uchar           sip_lsn;
uchar           x25d_lsn;
uchar           stats = 0;
uchar           loopback = 0;
uchar           gateway_up = 0;
uchar           xlisten = 0;
short           hangup;
short           isdn_packets_recv = 0;
short           isdn_bytes_recv = 0;
short           isdn_packets_sent = 0;
short           x25_packets_recv = 0;
short           x25_bytes_recv = 0;
short           x25_packets_sent = 0;
long            starttime=0;
ushort          remote_length;
ncb_t           iincb;          /* Input  ISDN NCB */
ncb_t           oincb;          /* Output ISDN NCB */
ncb_t           ixncb;          /* Input  X25  NCB */
ncb_t           oxncb;          /* Output X25  NCB */
ncb_t           lxncb;          /* Listen X25  NCB */

void send_cmd_60(ncb_t far *);
void send_cmd_5c(ncb_t far *);
void clear_ncb(ncb_t far *);
```

```
void interrupt far x25_listen_post();
void interrupt far post_rtn();
void (interrupt far *rtn_ptr)();
void x25_listen();
void isdn_x25d();
void isdn_sip();
void isdn_start_dpp();
void isdn_d_link_open();
void x25d_hangup();
void x25_hangup();
void isdn_call_setup(ncb_t far *);
void x25_call_wait(ncb_t far *);
void isdn_call_wait(ncb_t far *);
void interrupt far x25_send();
void interrupt far isdn_send();
void gateway_setup(ncb_t far *, ncb_t far *, ncb_t far *, ncb_t
 far *);
void check_call_sw(ncb_t far *, ncb_t far *, ncb_t far *, ncb_t
 far *);
void shutdown();
void main_menu();
void view_statistics();
void x25_loopback(ncb_t far *);
void trans_stats();
void x25_stats();
void isdn_stats();
void set_x25_address(uchar, char *, char *);
void output_message(char *);
void selection();
void stat_scr();
void x25_cancel();
void x25d_cancel();

main()
{
        uchar           ll;
        uchar           rl;
        ushort          setup_gateway = 0;
        ushort          i;
        char            ans;
        ncb_t far       *iinp;
        ncb_t far       *oinp;
        ncb_t far       *ixnp;
        ncb_t far       *oxnp;
        ncb_t far       *lxnp;   /* Listen X25 ncb */

        iinp = &iincb;
        oinp = &oincb;
        ixnp = &ixncb;
        oxnp = &oxncb;
        lxnp = &lxncb;
        i = 0;

        main_menu();
        hangup = 0;
```

```
ans = 0;
x25_call_sw = 0;

isdn_call_sw = 0;

while(ans!='9')
{
        if(kbhit()!=0)
        {
                ans = getche();
                if(menu_choice[(ans-49)]==1)
                {
                        sprintf(obuf, "Menu choice already
                        selected.\n");
                        output_message(obuf);
                }
                else
                {
                switch(ans)
                {
                case '1' :
                        system("X25_START");
                        menu_choice[0] = 1;
                        break;
                case '2' :
                        system("ISDN_START");
                        menu_choice[1] = 1;
                        break;
                case '3' :
                        x25_listen(lxnp);
                        setup_gateway = 1;
                        gateway_up = 1;
                        _settextposition(12,57);
                        printf("[ON] \n");
                        break;
                case '4' :
                        menu_choice[3] = 1;
                        _settextposition(13,57);
                        if(loopback==0)
                        {
                                loopback=1;
                                printf("[ON] \n");
                        }
                        else
                        {
                                loopback = 0;
                                printf("[OFF]\n");
                        }
                        break;
                case '5' :
                        shutdown();
                        for (i=0;i<8;i++)
                                menu_choice[i] = 0;
/*                       gateway_up = 0;     */
                        loopback = 0;
```

```
                        menu_choice[4] = 1;
                        break;
                case '6':
                        trans_stats();

                        break;
                case '7' :
                        x25_stats();
                        break;
                case '8' :
                        isdn_stats();
                        break;
                case '9':
                        _setvideomode(_DEFAULTMODE);
                        break;
                default :
                        sprintf(obuf,"Please stick to
                        reality...\n");
                        output_message(obuf);
                }
                }
                if((ans>'0')&&(ans<'3')||(ans>'4')&&
                (ans<'9'))main_menu();
        }
        if(setup_gateway==1)
        {
                if(xlisten!=0)
                {
                        sprintf(obuf,"X25 call established.
                        \n");
                        output_message(obuf);
                        setup_gateway = 2;
                        x25_lsn = lxnp->ncb_lsn;
                        ll = x25_buffer[0];
                        ll = (ll<<4);
                        ll = ll>>4;
                        rl = x25_buffer[0];
                        rl = rl>>4;
                        remote_length = ll;
                        set_x25_address(ll,x25_send_address,
                        &x25_buffer[1]);
                        set_x25_address(rl,x25_recv_address,
                        &x25_buffer[ll/2+1]);
                        gateway_setup(ixnp, oxnp, iinp,
                        oinp);
                        time(&starttime);
                }
        }
        check_call_sw(ixnp,oxnp,iinp,oinp);
        selection();
    }
/* End. */
}
```

```
void set_x25_address(uchar l, char *cp, char *ap)
{
        short i,j = 0;
        uchar t1,t2;

        i = 0;
        j = 1;
        while(i<l/2)
        {
                t1 = *ap;

                t2 = *ap++;
                t1 = (t1>>4);
                t2 = (t2<<4);
                t2 = (t2>>4);
                sprintf(cp++,"%c",(t1+48));
                sprintf(cp++,"%c",(t2+48));
                i++;
        }

}
/*
        Subroutine to acutally send out the command using the
        60 interrupt vector.
*/
void send_cmd_60(ncb_t far *ncbp)
{
        short ret       = 0;

        regs.x.bx       = FP_OFF(ncbp);
        sregs.es        = FP_SEG(ncbp);
        ret             = int86x(NETBIOS_60,&regs,&regs,&sregs);
/*
        if((ncbp->ncb_retcode>0)&&(ncbp->ncb_retcode!=0xff))
        {
        printf("ISDN : int86x retcode: 0x%x NCB retcode    : 0x%x ",
                ret&0x00ff,ncbp->ncb_retcode);
        printf("Session LSN    : 0x%x\n",ncbp->ncb_lsn);
        }
*/
}
/* Subroutine to clear out the NCB. */
void clear_ncb(ncb_t far *ncbp)
{
        int i;
        char far *cptr = (char far *) ncbp;
        for (i=0; i<sizeof(*ncbp); i++,cptr++)
                *cptr = '\x00';
}
void interrupt far post_rtn()
{
        sw = 1;
}
/*
        5C interrupt vector. */
```

```
void send_cmd_5c(ncb_t far *ncbp) {
        short ret       = 0;
        int i;
        ncb_t far *lp;
        regs.x.bx       = FP_OFF(ncbp);
        sregs.es        = FP_SEG(ncbp);
        lp = ncbp;

        ret             = int86x(NETBIOS_5C,&regs,&regs,&sregs); /*
        if((ncbp->ncb_retcode>0)&&(ncbp->ncb_retcode!=0xff)
        &&(ncbp->ncb_retcode!=0xa))
        {
        printf("\nX25  : int86x ret: 0x%x NCB ret   : 0x%x ",
                ret&0x00ff,ncbp->ncb_retcode);
        printf("Session LSN   : 0x%x\n",ncbp->ncb_lsn);
        } */
}
/*   Setup a LISTEN on the X25 line... */
void x25_listen(ncb_t far *xncbp)
{
        short i;

        clear_ncb(xncbp);
        xncbp->ncb_command = 0x91; /*NABIOS_CALL;*/
        destp = xncbp->ncb_callname;
        srcp = X25_NAME;
        for (i=0; i<16; i++, srcp++, destp++)
                *destp = *srcp;
        xncbp->ncb_buffer = x25_buffer;
        xncbp->ncb_lana_num = 0x0ff;
        xncbp->ncb_length = 7;

        rtn_ptr = x25_listen_post;
        xncbp->ui2 = FP_SEG(rtn_ptr);
        xncbp->ui1 = FP_OFF(rtn_ptr);

        xncbp->ncb_rto = 10;
        xncbp->ncb_sto = 10;
        memcpy(x25_buffer, call_buffer, 7);
        sprintf(obuf,"Waiting for X25 call...");
        output_message(obuf);
        send_cmd_5c(xncbp);
}

void interrupt far x25_listen_post()
/*
        Post routine to listen for X25 call.
*/
{ xlisten = 1; }

void isdn_x25d()



/* Call to *X25D to receive the lsn. */
```

```
{
        short i;
        ncb_t far *incbp;

        ncb_t ncb;

        incbp = &ncb;
        clear_ncb(incbp);
        incbp->ncb_command = 0x10;
        destp = incbp->ncb_callname;
        srcp = X25D_NAME;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
                *destp = *srcp;
        send_cmd_60(incbp);
        x25d_lsn = incbp->ncb_lsn; }

void isdn_sip()
/*
        Call to *SIP to receive its lsn.
*/
{
        short i;
        ncb_t far *incbp;
        ncb_t ncb;
        struct sip_setup
        {
                short sip_command;
                short reserved;
                short call_type;
                short r_adapt;
                short l2_spec;
                short l3_spec;
                short l2_comp;
                short l3_comp;
                short channel;
                short lsn;
                short dial_length;
                char  dial[4];
        }ss;

        struct sip_response
        {
                short sip_command;
                short reserved;
                short lsn;
                short return_code;
                short previous_command;
        }sr;

        incbp = &ncb;
        clear_ncb(incbp);
        incbp->ncb_command = 0x10;
```

```
        destp = incbp->ncb_callname;
        srcp = SIP_NAME;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
                *destp = *srcp;
        send_cmd_60(incbp);
        sip_lsn = incbp->ncb_lsn;
/* Call to *SIP to send setup info. */
        incbp->ncb_command = 0x14;
        incbp->ncb_lsn = sip_lsn;
        incbp->ncb_buffer = (char *) &ss;
        incbp->ncb_length = sizeof(ss);
        ss.sip_command = 0x01;
        ss.reserved = 0x0;
        ss.call_type = 0x01;
        ss.r_adapt = 0x00;
        ss.l2_spec = 0x06;
        ss.l3_spec = 0x06;
        ss.l2_comp = 0x06;
        ss.l3_comp = 0x06;
        ss.channel = 0x00;
        ss.lsn = (short) x25d_lsn;
        ss.dial_length = 0x0;
        sprintf(obuf,"SIP send setup.\n");
        output_message(obuf);
        send_cmd_60(incbp);
        sprintf(obuf,"SIP send setup return, dial length : %d\n",
        ss.dial_length);
        output_message(obuf);

/*
        Receive response from *SIP.
*/
        incbp->ncb_command = 0x15;
        incbp->ncb_lsn = sip_lsn;
        incbp->ncb_buffer = (char *) &sr;
        incbp->ncb_length = sizeof(sr);
        sprintf(obuf,"Sending SIP response.\n");
        output_message(obuf);
        send_cmd_60(incbp);
        sprintf(obuf,"SIP response received.\n");
        output_message(obuf);
/*
        Hangup to *SIP. */
        incbp->ncb_command = 0x12;
        incbp->ncb_lsn = sip_lsn;
        sprintf(obuf,"SIP hangup\n");
        output_message(obuf);
        send_cmd_60(incbp);
}
void isdn_start_dpp()
/*
        Send START_DPP to *X25D.
*/
{
```

```
        short i;
        ncb_t far *incbp;
        ncb_t ncb;
        struct iotcl_struct
        {

                short iotcl_cmd;
                short iotcl_cmd_cmd;
                short iotcl_cmd_arg[13];
        }iotcl;

        incbp = &ncb;
        rtn_ptr = post_rtn;       /* set the post routine pointer */
        clear_ncb(incbp);
        incbp->ncb_command = 0x14;   /* Send - 94 = no wait mode */
        destp = incbp->ncb_callname;
        srcp = X25D_NAME;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
                *destp = *srcp;
        incbp->ncb_lsn = x25d_lsn;
        incbp->ncb_buffer = (char *) &iotcl;
        incbp->ncb_length = sizeof(iotcl);
/*
        incbp->ui2 = FP_SEG(rtn_ptr);
        incbp->ui1 = FP_OFF(rtn_ptr);
*/
        iotcl.iotcl_cmd = 0x08;
        iotcl.iotcl_cmd_cmd = 0x01;
        iotcl.iotcl_cmd_arg[0] = 8;
        iotcl.iotcl_cmd_arg[1] = 0;
        iotcl.iotcl_cmd_arg[2] = 0;
        for(i=3;i<13;i++)
                iotcl.iotcl_cmd_arg[i] = 0;
        sw = 0;
        i = 0;
        sprintf(obuf,"Start_Dpp.\n");
        output_message(obuf);
        send_cmd_60(incbp);
/*
        while(sw==0)
        {
                i = i;
        }
*/
        sprintf(obuf,"Layers 2 and 3 are up...\n");
        output_message(obuf);
}
void isdn_d_link_open()
/*
        Send a LINK_OPEN message to *X25D.
*/
{
        short i;
        short st;
        short ad;
```

```
short xcall;
short tr9999;
short tr0144;
short tr0044;
short ts0145;
short ts0144;

short dl;
ncb_t far *incbp;
ncb_t ncb;
struct link_struct
{
        short link_op_cmd;
        short link_op_size;
        char link_op_control[64];
}link;
struct dpp_response
{
        short link_op_res;
        short link_op_ret_code;
        short link_op_LID;
        short link_op_PCC;
        short link_op_cause;
        short link_op_diag;
        short link_op_session;
}r;

incbp = &ncb;
clear_ncb(incbp);
destp = incbp->ncb_callname;
srcp = X25D_NAME;
for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
        *destp = *srcp;
incbp->ncb_command = 0x14;
incbp->ncb_lsn = x25d_lsn;
incbp->ncb_buffer = (char *) &link;
incbp->ncb_length = sizeof(link);
link.link_op_cmd = 0x05;
link.link_op_size = 64;
for (i=0; i<64; i++)
        link.link_op_control[i] = 0x0;
st = 14;

ad = 0;
tr9999 = strncmp(&x25_recv_address[6],"9999",4);
tr0144 = strncmp(&x25_recv_address[6],"0144",4);
tr0044 = strncmp(&x25_recv_address[0],"804",3);
ts0145 = strncmp(&x25_send_address[6],"0145",4);
ts0144 = strncmp(&x25_send_address[6],"0144",4);
xcall = 0; /*
_settextposition(2,1);
printf(" 999 : %d 144 : %d 044 : %d\n",tr9999,tr0144,tr0044);
printf(" 145 : %d 144 : %d\n",ts0145,ts0144);
*/
if(tr9999==0)
```

```
{
        st = 14;
        dl = 4;
        if(remote_length>4)
        {
                st = 19;

                memcpy(&link.link_op_control[14],X25_DIAL_COD
                dl = 15;
        }
        memcpy(&link.link_op_control[st],x25_send_address,
        remote_length); /* target address */
        link.link_op_control[st+remote_length] = 0x0;
        if(remote_length>4)xcall=1;
}
if(tr0144==0)
{
        memcpy(&link.link_op_control[14],"1850",4);
        /* target address */
        link.link_op_control[18] = 0x0;
        dl = 4;
}
if(tr0044==0)
{
        if(ts0145==0)
        {
                memcpy(&link.link_op_control[14],"1850",4);
                /* target address */
                link.link_op_control[18] = 0x0;
                dl = 4;
        }
        else
        {
                memcpy(&link.link_op_control[14],
                "031418846440145",15); /* target address */
                link.link_op_control[29] = 0x0;
                xcall = 1;
                dl = 15;
        }
}
if(xcall==1)
{
        link.link_op_control[0] = 0x0080;
        link.link_op_control[48] = 1;
        link.link_op_size = 51;
}
memcpy(&link.link_op_control[30],"0002",4);
/* source address */
link.link_op_control[34] = 0x0;
link.link_op_control[46] = 0x1;
sprintf(obuf,"Calling :%s",link.link_op_control[14]);
output_message(obuf);
send_cmd_60(incbp);
for(i=0;i<dl;i++)
        isdn_send_address[i] = link.link_op_control[14+i];
```

```
        for(i=0;i<5;i++)
                isdn_recv_address[i] = link.link_op_control[30+i];
/*
        Receive RESPONSE back from *X25D.
*/
        clear_ncb(incbp);
        destp = incbp->ncb_callname;
        srcp = X25D_NAME;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
                *destp = *srcp;
        incbp->ncb_command = 0x15;
        incbp->ncb_lsn = x25d_lsn;

        incbp->ncb_buffer = (char *) &r;
        incbp->ncb_length = sizeof(r);
        send_cmd_60(incbp);
        lid = r.link_op_LID;
        sprintf(obuf,"X25 - ISDN connection established.\n");
        output_message(obuf);
/*
        printf("Start_DPP return code: 0x%x\n",r.link_op_ret_code);
        printf("Link identifier      : %d\n",r.link_op_LID);
        printf("Session Number       : 0x%x\n",r.link_op_session);
*/
}

void x25d_hangup()
/*
        Hangup the *X25D.
*/ {
        short i;
        ncb_t far *incbp;
        ncb_t ncb;

        incbp = &ncb;
        clear_ncb(incbp);
        destp = incbp->ncb_callname;
        srcp = X25D_NAME;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
                *destp = *srcp;
        incbp->ncb_command = 0x12;
        incbp->ncb_lsn = x25d_lsn;
        send_cmd_60(incbp);
        sprintf(obuf,"ISDN D channel hangup.\n");
        output_message(obuf);
}

void x25_hangup()
/*
        Hangup the X25 channel.
*/ {
        short i;
        ncb_t far *xncbp;
        ncb_t ncb;
        char    hang_buffer[256];
```

```
        xncbp = &ncb;
        clear_ncb(xncbp);
        destp = xncbp->ncb_callname;
        srcp = X25_NAME;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
                *destp = *srcp;
        xncbp->ncb_command = 0x12;
        xncbp->ncb_lana_num = 0x0ff;
        xncbp->ncb_lsn = x25_lsn;
        xncbp->ncb_buffer = hang_buffer;

        xncbp->ncb_length = 0;
        send_cmd_5c(xncbp);
        sprintf(obuf,"X25 hangup.\n");
        output_message(obuf);
}

void isdn_call_setup(ncb_t far *oinp) /*
        Setup the output to ISDN X25D passing structure.
*/ {
        short i;

        clear_ncb(oinp);
        destp = oinp->ncb_callname;
        srcp = X25D_NAME;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
                *destp = *srcp;
        oinp->ncb_command = 0x14;
        oinp->ncb_lsn = x25d_lsn;
        oinp->ncb_buffer = (char *) &ou.dm;
        oinp->ncb_length = sizeof(ou.dm);
        ou.dm.data_cmd = 0x01;
        ou.dm.data_LID = lid;
        ou.dm.data_Control = 0x0;
}

void x25_call_setup(ncb_t far *oxnp)
/*
        Setup the output to X25 passing structure.
*/
{
        short i;

        clear_ncb(oxnp);
        destp = oxnp->ncb_callname;
        srcp = X25_NAME;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
                *destp = *srcp;
        oxnp->ncb_command = 0x14;
        oxnp->ncb_lsn = x25_lsn;
        oxnp->ncb_buffer = iu.dm.data_Data;
        oxnp->ncb_lana_num = 0x0ff;
}
```

```
void x25_call_wait(ncb_t far *ixnp)
/*
        Setup Wait routine to receive data from the remote X25 site.
*/
{
        short i;

        rtn_ptr = x25_send;
        clear_ncb(ixnp);
        srcp = X25_NAME;

        destp = ixnp->ncb_callname;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
                *destp = *srcp;
        ixnp->ncb_command = 0x95;
        ixnp->ncb_lsn = x25_lsn;
        ixnp->ncb_lana_num = 0x0ff;
        ixnp->ncb_buffer = ou.dm.data_Data;
        ixnp->ncb_length = 128;

        ixnp->ui2 = FP_SEG(rtn_ptr);
        ixnp->ui1 = FP_OFF(rtn_ptr);

        send_cmd_5c(ixnp);
}

void isdn_call_wait(ncb_t far *iinp)
/*
        Setup Wait routine to receive data from the ISDN X25D site.
*/
{
        short i;

        rtn_ptr = isdn_send;
        clear_ncb(iinp);
        srcp = X25D_NAME;
        destp = iinp->ncb_callname;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
                *destp = *srcp;
        iinp->ncb_lsn = x25d_lsn;
        iinp->ncb_buffer = (char *) &iu.dm;
        iinp->ncb_length = sizeof(iu.dm);
        iu.dm.data_cmd = 0x01;
        iu.dm.data_LID = lid;
        iu.dm.data_Control = 0x0;
        iinp->ncb_command = 0x95;

        iinp->ui2 = FP_SEG(rtn_ptr);
        iinp->ui1 = FP_OFF(rtn_ptr);

        send_cmd_60(iinp);
}
void interrupt far isdn_send()

/*      ISR for ISDN X25 D channel send. */
```

```
{
        isdn_call_sw = 1; }
void interrupt far x25_send()
/*
        ISR for X25 send.
*/
{
        x25_call_sw = 1; }

void gateway_setup(ncb_t far *ixnp, ncb_t far *oxnp,
                   ncb_t far *iinp, ncb_t far *oinp)
/*
        Setup the gateway for transmissions.
*/
{
        if(loopback==0)
        {       isdn_x25d();
                isdn_sip();
                isdn_start_dpp();
                isdn_d_link_open();
                isdn_call_setup(oinp);
                isdn_call_wait(iinp);
        }
        x25_call_setup(oxnp);
        x25_call_wait(ixnp);
}
void check_call_sw(ncb_t far *ixnp, ncb_t far *oxnp,
                   ncb_t far *iinp, ncb_t far *oinp)
/*
        Subroutine to check for a call appearance
*/
{
        short   update_needed;
        short   i;
        short   nf;
        short   hup = 0;

        update_needed = 0;
        if(x25_call_sw!=0)
        {
                if(loopback!=0)
                        x25_loopback(ixnp);
                else
                {
                        if(stats!=0)update_needed = 1;
                        if(ixnp->ncb_length>0)
                        {
                                ou.dm.data_Size = ixnp->ncb_length;
                                send_cmd_60(oinp);
                        }
                }
                if((ixnp->ncb_retcode==0x0a)&&(hangup==0))
                {
                        if(loopback==0)x25d_hangup();
                        sprintf(obuf,"X25 hangup received.\n");
```

```
                        output_message(obuf);
                        hangup = 1;
                }
                else
                {
                        x25_bytes_recv += ixnp->ncb_length;
                        x25_packets_recv++;
                        x25_call_wait(ixnp);
                        x25_packets_sent++;

                }
                x25_call_sw = 0;
        }
        if(isdn_call_sw!=0)
        {
                if(stats!=0)update_needed = 1;
                switch (iu.dm.data_cmd)
                {
                        case 1 :
                                if(iu.dm.data_Size>0)
                                {
                                        oxnp->ncb_buffer
                                        = iu.dm.data_Data;
                                        oxnp->ncb_length
                                        = iu.dm.data_Size;
                                        send_cmd_5c(oxnp);
                                }
                                break;
                        case 2 :
                                nf = 0;
                                i = 0;
                                while((i<128)&&(nf==0))
                                {
                                        if(iu.display.dis_Message[i]
                                        ==0x0)
                                        {
                                                oxnp->ncb_length = i;
                                                nf = 1;
                                        }
                                }
                                oxnp->ncb_buffer =
                                iu.display.dis_Message;
                                send_cmd_5c(oxnp);
                                break;
                        case 3 :
                                if((iu.ioctl.ioctl_CODE>2)&
                                (hangup==0))
                                        hup = 1;
                                else
                                        sprintf(obuf,
                                        "Unrecognized ioctl code :
                                        %d\n",iu.ioctl.ioctl_CODE);
                                        output_message(obuf);
                                break;
                }
```

```
                if(hup!=0)
                {
                        x25_hangup();
                        sprintf(obuf,"ISDN hangup received.\n");
                        output_message(obuf);
                        hangup = 1;
                }
                else
                {
                        isdn_bytes_recv +=ixnp->ncb_length;
                        isdn_packets_recv++;
                        isdn_packets_sent++;
                        isdn_call_setup(oinp);
                        isdn_call_wait(iinp);
                }
                isdn_call_sw = 0;
        }

}

void shutdown() {
        if(gateway_up==1)
        {
                if(loopback==0)
                {
                        isdn_call_sw = 0;
                        x25d_hangup();
/*                       x25d_cancel();   */
                }
                x25_call_sw = 0;
                x25_hangup();
/*               x25_cancel();    */
        }
        hangup = 1;
        sprintf(obuf,"Gateway is shutdown.\n");
        output_message(obuf);
}

void x25_cancel()
{
        ncb_t far *cnp;
        ncb_t ncb;
        int i;

        clear_ncb(cnp);
        cnp = &ncb;
        cnp->ncb_command = 0x35;
        cnp->ncb_length = 0;
        cnp->ncb_buffer = (char *) &ixncb;
        destp = cnp->ncb_callname;
        srcp = X25_NAME;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
                *destp = *srcp;
        cnp->ncb_lsn = x25_lsn;
        cnp->ncb_lana_num = 0xff;
```

```
        send_cmd_5c(cnp);
}

void x25d_cancel()
{
        ncb_t far *cnp;
        ncb_t ncb;
        int i;

        clear_ncb(cnp);
        cnp = &ncb;
        cnp->ncb_command = 0x35;
        cnp->ncb_length = 0;
        cnp->ncb_buffer = (char *) &iincb;
        destp = cnp->ncb_callname;
        srcp = X25D_NAME;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)

                *destp = *srcp;
        cnp->ncb_lsn = x25d_lsn;
        send_cmd_60(cnp);
}

void main_menu()
/*
        Main Menu Screen routine.
*/
{
        struct videoconfig vc;

/*      SET BACKGROUND */
        if (_setvideomode(_ERESCOLOR) == 0) {
                printf ("%s\n", error_message);
                exit(0);
        }
        _getvideoconfig (&vc);
        _setbkcolor(_BLUE);

/*      ADD TOP RECTANGLE */
        _setlogorg (vc.numxpixels/2-1,0);
        _setcolor(15);
        _setlinestyle(0xFFFF);
        _rectangle(_GBORDER, -300,0,300,40);
        _setcolor(9);
        _rectangle(_GBORDER, -299,1,299,39);

/*      ADD TEXT
*/
        _settextcolor (10);
        _settextposition (2,30);
        sprintf(buffer, "X.25 - ISDN GATEWAY v1.0\n");
        _outtext (buffer);
        _settextcolor (15);
        _settextposition (6,37);
        sprintf(buffer, "MAIN MENU \n");
```

```
        _outtext (buffer);
        printf("\n\t\t\t    Gateway Menu Options: \n\n");
        printf("\t\t\t    1)  Reset and Load X.25 \n");
        printf("\t\t\t    2)  Reset and Load ISDN \n");
        printf("\t\t\t    3)  Execute Gateway Startup ");
        _settextposition(12,57);
        if(gateway_up==0)
                printf("[OFF]\n");
        else
                printf("[ON] \n");
        printf("\t\t\t    4)  Toggle X25 Loopback     ");
        _settextposition(13,57);
        if(loopback==0)
                printf("[OFF]\n");
        else
                printf("[ON] \n");

        printf("\t\t\t    5)  Clear Calls\n");
        printf("\t\t\t    6)  View Transmission Statistics\n");
        printf("\t\t\t    7)  View X25  Protocol Statistics\n");
        printf("\t\t\t    8)  View ISDN Protocol Statistics\n");
        printf("\t\t\t    9)  Exit to DOS\n\n");

        _rectangle(_GBORDER, -300,44,300,296);
        _rectangle(_GBORDER, -299,45,299,295);

/*      ADD LOWER RECTANGLE
*/
        _setcolor(7);
        _setlinestyle(0xFFFF);
        _rectangle(_GBORDER, -300,300,300,340);
        _rectangle(_GBORDER, -299,301,299,339);
        _settextcolor (3);
        _settextposition (20,28);
        sprintf(buffer, " Please enter a selection:  ");
        _outtext (buffer);
        selection();
}

void selection() {
        _settextposition(20,56);
        printf("   ");
        _settextposition(20,56); }
void x25_loopback(ncb_t far *incb) /*
        Subroutine to perform a loopback on the X25 circuit.
*/ {
        ncb_t far *oncb;
        ncb_t ncb;
        short i;

        oncb = &ncb;
        clear_ncb(oncb);
        destp = oncb->ncb_callname;
        srcp = X25_NAME;
        for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
```

```
                *destp = *srcp;
        oncb->ncb_command = 0x14;
        oncb->ncb_lsn = x25_lsn;
        oncb->ncb_buffer = incb->ncb_buffer;
        oncb->ncb_length = incb->ncb_length;
        oncb->ncb_lana_num = 0x0ff;
        send_cmd_5c(oncb);
}

void trans_stats()
{
        long curtime;
        int i;


        time(&curtime);

        _clearscreen(_GCLEARSCREEN);
        _settextcolor (15);
        _settextposition (6,28);
        sprintf(obuf, "GATEWAY STATISTICS SCREEN \n");
        _outtext (obuf);
        if(starttime>0) {_settextposition(5,60);
                         sprintf(obuf,"Up Time: %d secs",
                        curtime-starttime);
                         _outtext(obuf);
                        }
        _settextposition (8,1);
        printf("\n\tRemote 1 [X25] \t\t\tRemote 2 ");
        if(loopback==0)
                printf("[ISDN]");
        else
                printf("[Loopback Mode]");
        _settextposition(10,1);
        printf("\n\tSend Address : \t\t\tSend Address :\n");
        printf("\tRecv Address : \t\t\tRecv Address : \n");
        printf("\tPackets Recv : \t\t\tPackets Recv : \n");
        printf("\tBytes Recv   : \t\t\tBytes Recv   : \n");
        printf("\tPackets Sent : \t\t\tPackets Sent : \n");
        _settextposition(11,24);sprintf(obuf,"%s",x25_send_address);
        _outtext(obuf);
        _settextposition(11,56);sprintf(obuf,"%s",isdn_send_address);
        _outtext(obuf);
        _settextposition(12,24);sprintf(obuf,"%s",x25_recv_address);
        _outtext(obuf);
        _settextposition(12,56);sprintf(obuf,"%s",isdn_recv_address);
        _outtext(obuf);
        _settextposition(13,24);sprintf(obuf,"%d",x25_packets_recv);
        _outtext(obuf);
        _settextposition(13,56);sprintf(obuf,"%d",isdn_packets_recv);
        _outtext(obuf);
        _settextposition(14,24);sprintf(obuf,"%d",x25_bytes_recv);
        _outtext(obuf);
        _settextposition(14,56);sprintf(obuf,"%d",isdn_bytes_recv);
        _outtext(obuf);
```

```
        _settextposition(15,24);sprintf(obuf,"%d",x25_packets_sent);
        _outtext(obuf);
        _settextposition(15,56);sprintf(obuf,"%d",isdn_packets_sent);
        _outtext(obuf);
        _settextposition(17,9);
        printf("Remote 1 Last Packet Received :");
        if(loopback!=0)
        {
                for (i=0;i<ixncb.ncb_length;i++)printf("%c",
                ou.dm.data_Data[i]);
        }
        else
        {
                for(i=0;i<iu.dm.data_Size;i++)printf("%c",iu.dm.data_
                Data[i]);
        }
        _settextposition(19,9);
        printf("Remote 2 Last Packet Received :");
        if(loopback==0)
        {
                for(i=0;i<ou.dm.data_Size;i++)printf("%c",ou.dm.data_
                Data[i]);
        }
        stat_scr();
        sprintf(obuf,"Hit ENTER to return to main menu.\n");
        output_message(obuf);
        getch();
}


void x25_stats() {
        int i;
        ncb_t ncb;
        ncb_t far *sncb;
        struct xstat_record
        {
                ushort  reserve;
                ushort  lcn;
                uchar   session;
                uchar   status;
                char    aname[16];
                char    pname[16];
                uchar   rncbp;
                uchar   sncbp;
                char    DTE[16];
                ulong   npos;
                ulong   npya;
                ulong   ncos;
                ulong   ncya;
                ulong   nprn;
                ulong   npyr;
                ulong   ncrn;
                ulong   ncyr;
                ulong   nrDTE;
                ulong   nrnet;
```

```
}xstat;

clear_ncb(sncb);
sncb->ncb_command = 0x34;
sncb->ncb_lsn = x25_lsn;
sncb->ncb_buffer = (char *) &xstat;
sncb->ncb_length = sizeof(xstat);
sncb->ncb_lana_num = 0xff;
srcp = X25_NAME;
destp = sncb->ncb_callname;
for(i=0;i<NB_NAME_LEN;i++,srcp++,destp++)
        *destp = *srcp;
send_cmd_5c(sncb);

_clearscreen(_GCLEARSCREEN);
_settextcolor (15);
_settextposition (5,30);
sprintf(obuf,"X25 STATISTICS SCREEN \n");
_outtext(obuf);
_settextposition(8,17);
printf("Session number                    :
%d\n",xstat.session);
printf("\t\tSession status                    : ");
switch (xstat.status)
{
        case 1 :
                printf("LISTEN pending\n");
                break;
        case 2 :

                printf("CALL pending\n");
                break;
        case 3 :
                printf("Session connected\n");
                break;
        case 4 :
                printf("HANGUP pending\n");
                break;
        case 5 :
                printf("HANGUP complete\n");
                break;
        case 6 :
                printf("Circuit Cleared\n");
                break;
}
printf("\t\tNumber of Receive NCBs pending    :
%d\n",xstat.rncbp);
printf("\t\tNumber of Send    NCBs pending    :
%d\n",xstat.sncbp);
printf("\t\tUser packets offered/not acked    :
%d/%d\n",
        xstat.npos,xstat.npya);
printf("\t\tUser characters offered/not acked : %d/%d\n",
        xstat.ncos,xstat.ncya);
printf("\t\tNetwork packets offered/not read  : %d/%d\n",
```

```
                xstat.nprn,xstat.npyr);
        printf("\t\tNetwork character offered/not read: %d/%d\n",
                xstat.ncrn,xstat.ncyr);
        printf("\t\tNumber of resets by local DTE     : %d\n",
                xstat.nrDTE);
        printf("\t\tNumber of resets by the Network   : %d\n",
                xstat.nrnet);
        stat_scr();
        sprintf(obuf,"Hit ENTER to return to main menu.\n");
        output_message(obuf);
        getch();
}
void isdn_stats() {
        _clearscreen(_GCLEARSCREEN);
        _settextcolor (15);
        _settextposition (5,30);
        sprintf(obuf,"ISDN STATISTICS SCREEN \n");
        _outtext(obuf);
        _settextposition(8,17);
        printf("Option not currently available.\n");
        stat_scr();
        sprintf(obuf,"Hit ENTER to return to main menu.\n");
        output_message(obuf);
        getch();
}

void output_message(char *bufptr)
{
        char blanks[71];
        int     i, init=0;

        if(init==0)

        {
                for(i=0; i<71; i++)
                        blanks[i] = ' ';
                blanks[71] = 0x0;
                init = 1;
        }
        _settextposition (23,7);
        _outtext (blanks);
        _settextposition (23,7);
        _outtext (bufptr);
}
void stat_scr()
{
        struct videoconfig vc;
        long int j = 7;

/*  ADD TOP RECTANGLE */

        _setcolor(15);
        _setlinestyle(0xFFFF);
        _rectangle(_GBORDER, -300,0,300,40);
        _setcolor(9);
```

```
            _rectangle(_GBORDER, -299,1,299,39);

    /*  ADD TEXT */

            _settextcolor (10);
            _settextposition (2,32);
            sprintf(buffer, "X.25 - ISDN GATEWAY \n");
            _outtext (buffer);

            _rectangle(_GBORDER, -300,44,300,296);
            _rectangle(_GBORDER, -299,45,299,295);

    /* ADD LOWER RECTANGLE */

            _setcolor(7);
            _setlinestyle(0xFFFF);
            _rectangle(_GBORDER, -300,300,300,340);
            _rectangle(_GBORDER, -299,301,299,339);

    }
```

# 4.0 SUMMARY

In this document we presented the detailed software design for the gateway. A listing of the optimized software code is also presented. The code was optimized to provide minimum call setup delay, maximum throughput, and efficient information transfer. The code was developed in the "C" programming language using Microsoft Quick C V1.1 and the Microsoft Quick C Compiler for the DOS environment.

After testing and demonstration at the C&P site in Richmond, Va., it was determined that the gateway hardware and software operated efficiently and performed as expected. At the BAH System Resource Center in Bethesda, MD, the gateway and its associated software has also been tested and demonstrated using our Teleos IAP ISDN switch. Additional changes were made to make the code portable across platforms and to support other ISDN-type switches. Minor modifications to the code are necessary for the gateway to support the various ISDN switches.

The gateway program currently runs with full EGA graphics. However, it can be easily modified to run with monochrome systems. The program can be run from a floppy disk as well as a hard disk. The executable file is approximately 13,000 bytes. The gateway hardware can also be used as an ISDN enduser terminal or an X.25 enduser terminal providing that the appropriate software is loaded.

The next step in the ISDN gateway development will be to develop and implement a gateway that will allow X.25 and DDN (TCP/IP protocol) enduser terminals to interconnect through the ISDN D channel. The gateway should be able to automatically identify the protocol of the frame being received and switch to the proper transformation and mapping routines.